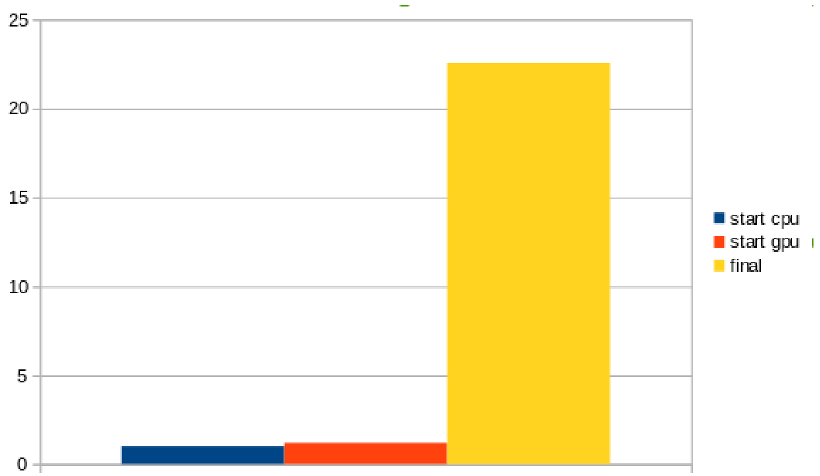
Bingo + SMCPy

BingoBros, D309/NASA Langley

Application Background

- Combination of symbolic regression (i.e., interpretable machine learning) and uncertainty quantification
- Enables learning analytical models from noisy, real-world data
- Large number of embarrassingly parallel computations
- Research powering the code is currently being funded by IRAD for FY22
- Code is open source and currently being used by a number of universities and NASA partners



23x speedup gained in hackathon!

Hackathon Objectives and Approach

- Push equation evaluation to GPU by parallelizing across different parameters and batching equations
- Utilize cupy as a drop-in replacement for numpy vectorization
- Analyze performance bottlenecks using nsys and nsight compute profiling
- Refactor algorithms to be more amenable to GPU programming

Technical Accomplishments and Impact

- *Speedup compared to:*

  CPU implementation: **23x**

  Initial GPU implementation: **10x?**
- *How did you achieve it?*
  GPU parallelization of the primary computations (across all models, constants and data)
- *Why does it matter / what does it enable?*
  Original combination of codes resulted in significant slowdown;
  GPU version makes the novel machine learning method tractable
- *Plan to continue work?* Yes! Continued GPU access will be critical.

Application Background

- DELTA is an open-source framework written to simplify deep learning with satellite imagery
- We leveraged multiple GPUs and multiple nodes with GPUs to attain faster training times
- Will benefit users of DELTA: scientists working with Earth science imagery and ML

	1 GPU	4 GPUs	Multi-Node
Epoch Train Time	2800 s	1500 s (1.85x)	TBD

Hackathon Objectives and Approach

- Python/Tensorflow/Tensorboard
- GDAL, common python libraries
- Profiling to identify data pipeline hotspots
- Comparing epoch training times across different resource levels

Technical Accomplishments and Impact

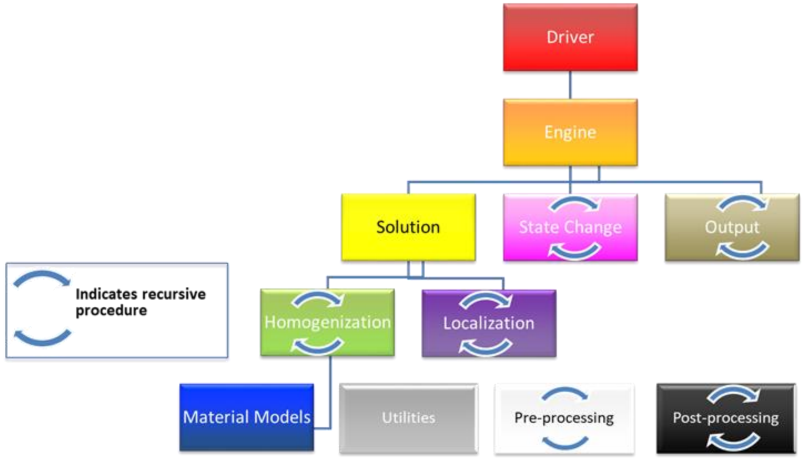
- We were able to profile our application with several different GPU resource levels
- We achieved multi-node training on a toy problem with progress on implementing this in production code
- This will reduce training times for some model architecture and dataset types
- After the hackathon we will be finishing production code integration and testing

Exploring GPU Parallelism for NASMAT

NASA Multiscale Analysis Tool (NASMAT), NASA GRC/LMS

Application Background

- A robust, modular tool for performing multiscale analyses of materials
- Next evolution of NASA MAC/GMC software
- Computational motifs targeted at hackathon - Sparse/Dense Linear Algebra, Structured Grids
- Stakeholders – ARMD/TTT, STMD/ESM



NASMAT's structure in terms of core procedures – A code re-write is necessary to improve parallel performance

Hackathon Objectives and Approach

- Programming models – Fortran, OpenACC
- Profiling / hot spots – Homogenization/Localization
- Libraries – Intel MKL, NVTX
- Performance tuning – Parallelizing highest level

Technical Accomplishments and Impact

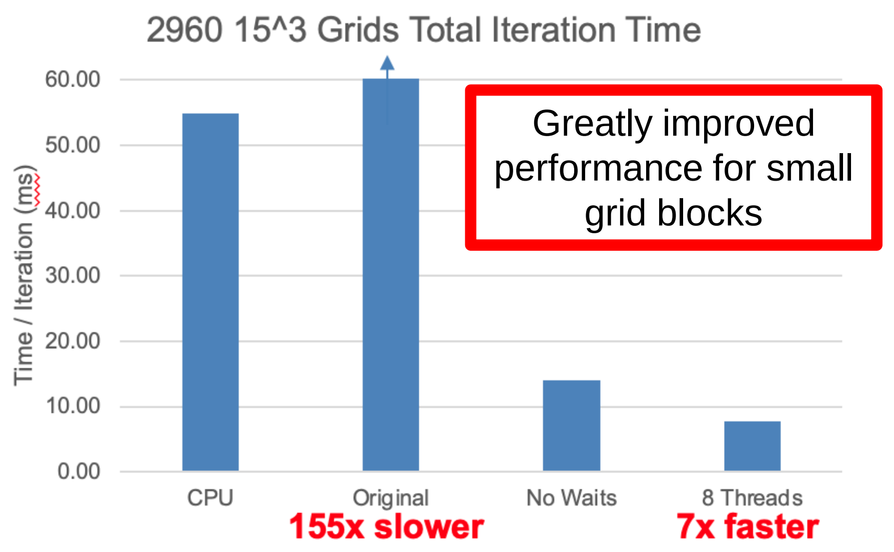
- Able to definitively determine that a code rewrite was needed
- Learned the basics of OpenACC and was able to successfully implement them in the code
- Saw some improvements in speed, but plenty more to work on
- **This event has helped the team get a better idea of how to proceed over the next year of code development.**

OVERFLOW

Central solver Mini-App / NASA Langley

Application Background

- OVERFLOW is a widely used CFD application using structured, overset grids to simulate a wide variety of problems in government, industry, and academia.
- During this hackathon we focused on some basic routines that represent a lot of the data access and computational patterns in the main application.
- Structured, overset, finite difference solver



Hackathon Objectives and Approach

- OpenACC+CudaFortran+CudaC+CuSparse
- Improve performance of some kernels through merging (improving arithmetic intensity) and reordering of tasks (load everything well before use to avoid long scoreboard waits)
- Rethinking our approach to parallelism by potentially looping over the grids inside the kernels to expose more parallelism

Technical Accomplishments and Impact

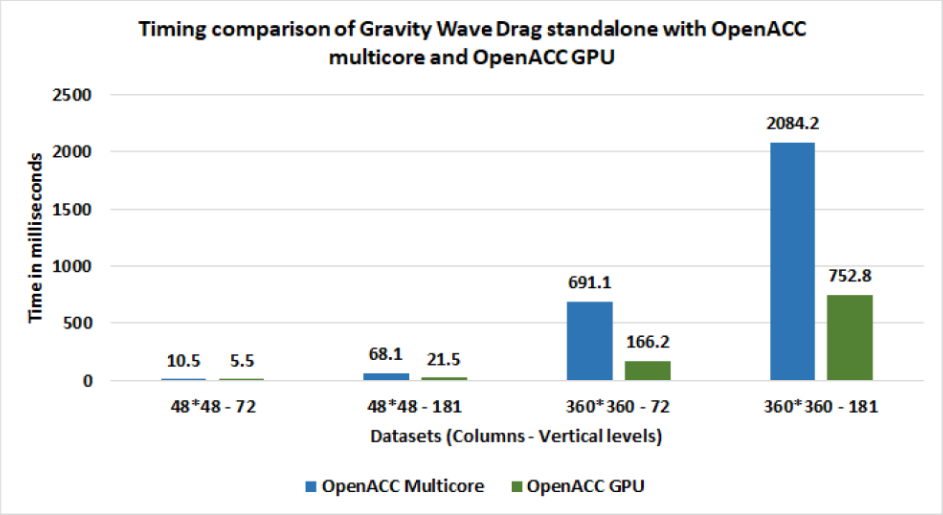
- We were able to speed up several kernels and improve the launch characteristics
- Avoiding excess waits, utilizing cuda streams, launching kernels in parallel
- Up to 7x times faster than CPU
- Enables simulations to be run faster (database generation, large cases)
- We have several plans and milestones to continue this work and would like continued support/access to the hardware from NASA

Physics Routine Porting to GPUs

Gravity Wave Drive / Shallow Convection / NASA Goddard

Application Background

- GEOS: Fortran-based coupled ocean-atmospheric model
- Gravity Wave Drive and Shallow Convection are part of GEO's parameterized physics routine codebase



Hackathon Objectives and Approach

- Gain experience using Kokkos and OpenACC
- Learn about profiling tools

Technical Accomplishments and Impact

- Created working standalones of codebases that utilize Kokkos and OpenACC
- Initial timings show the GPU executing 2-10x faster than the CPU code
- The work is a starting point for an effort to have the GEOS code running on GPUs

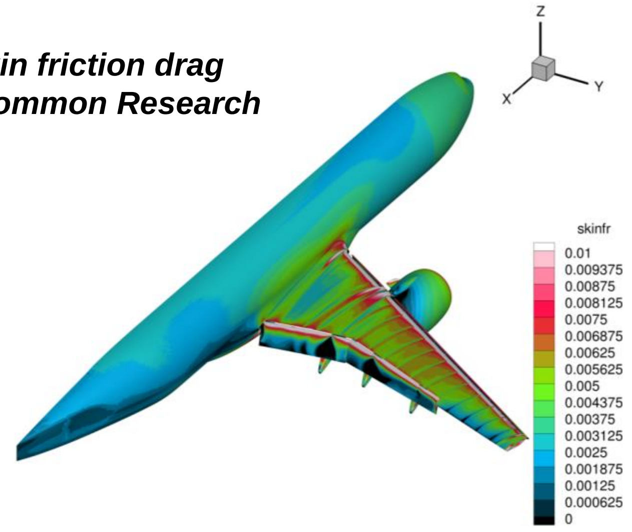
SLAT

Sparse Linear Algebra Toolkit / NASA Langley

Application Background

- Linear solver capabilities for Computational Aerodynamics
- Algorithmic building blocks and complete utilities supporting research and production
- Sparse Linear Algebra motif
- NASA and Aerospace industry partners

Prediction of skin friction drag forces on the Common Research Model



Hackathon Objectives and Approach

- Programming models
- - cuda, OpenACC
- Profiling / hot spots
- - ILU(k) solves
- Libraries
- - cuSparse and cuBLAS
- Performance tuning
- - cudaMemPrefetchAsync

Technical Accomplishments and Impact

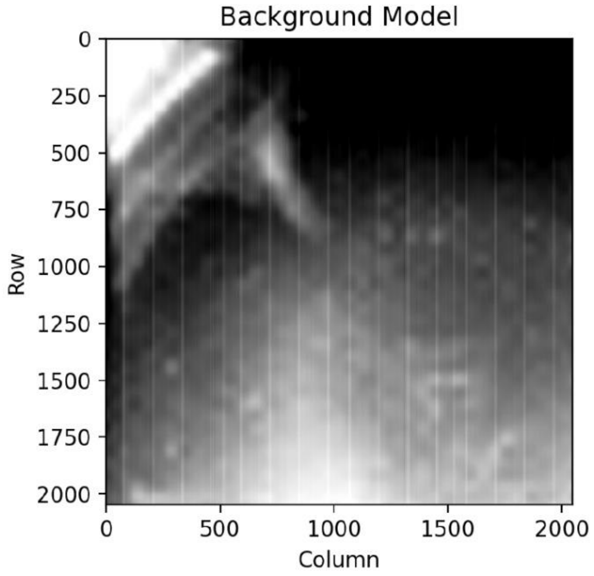
- 16x speed up over multi-core CPU
- Implemented Linear solver
- - ILU(k) preconditioning
- - GMRES
- Eddy-resolving methods for certification by analysis
- Will be integrated into Stabilized Finite Elements Library within FUN3D

tess-backdrop / scatterbrain

tess-backdrop / Code S, NASA Ames

Application Background

- We have an app *"tess-backdrop"* which builds a simple linear model for the scattered light background in TESS images.
- This model can be built for small patches very cheaply, after we first find and fix the weights.
- We wanted to run the first weight fit on the supercomputer, to process the images more quickly. Started with ~1 hour to process a single CCD/Sector of TESS data.



Hackathon Objectives and Approach

- We developed a "mini-app" which has only the core functionality of tess-backdrop; **scatterbrain**
- Python code
- Original code was fully numpy/CPU, we have updated the mini-app to allow either CPU or GPU computing.

Technical Accomplishments and Impact

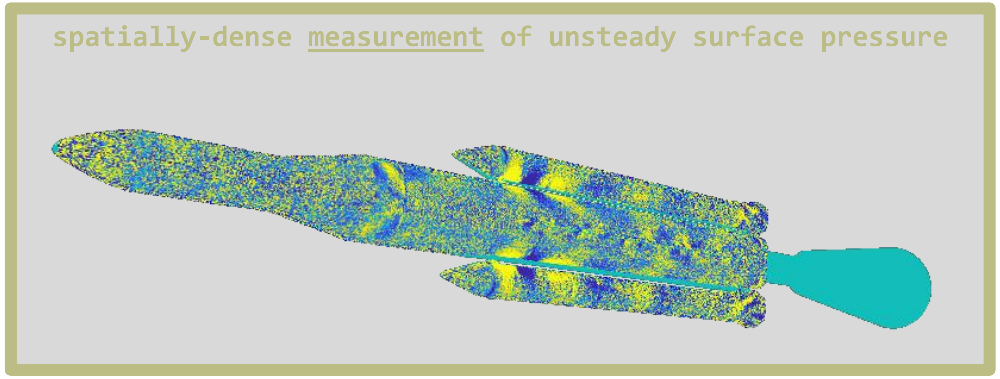
- We were able to achieve a >40x speedup over CPU, which greatly improves what we will be able to achieve with our main tool.
- We have learned cupy and MPI to obtain these speed ups and will be implementing the changes in our full application after the hackathon.
- We've learned how to profile our new GPU code with nsight

Team UPSP

Unsteady Pressure Sensitive Paint / NASA Ames

Application Background

- Unsteady Pressure Sensitive Paint: Wind tunnel surface pressure measurements with unprecedented spatial and temporal resolution.
- Also generates unprecedented volumes of data, sent to NAS during test for near real-time processing.
- Current application: launch vehicle aeroacoustics and buffet (SLS)
- Currently all CPU based. Team is unfamiliar with GPUs.



Hackathon Objectives and Approach

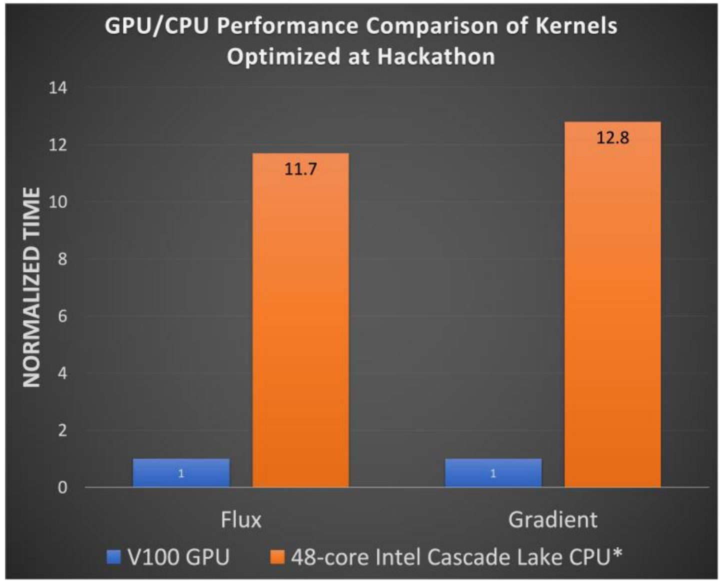
- Improve small step in uPSP software pipeline (spectral analysis)
- Brought a standalone mini-app to the event (separate code base, build system, etc)
- Practice implementing “bare” CUDA in C to learn more about details of CPU/GPU communication and coordination

Technical Accomplishments and Impact

- Accomplishments: Improved upsp_fft_decomposition. Used profiler, implemented GPU speedups, current DMD algorithm
- 15 seconds down to 5 seconds including GPU, CPU, IO
- Build tribal knowledge about GPUs. Successful team training exercise
- Miniapp was small aspect of our project. Will apply lessons to other parts of uPSP software.

Application Background

- Hypersonic Computational Fluid Dynamics Simulator.
- Focused on air-breathing scramjet / ramjet hypersonic vehicles.
- Stakeholders:
- NASA Hypersonic Technology Project
- DoD
- Industry partners
- University partners



Hackathon Objectives and Approach

- Port Fortran90 to C++ and Kokkos
- Ported and profiled unstructured inviscid residual, gradient evaluation, and inviscid Jacobian calculation.
- Performance tuned the residual and gradient kernels.

Technical Accomplishments and Impact

- Achieved an estimated ~12x speed up for targeted kernels compared to CPU based compute node*
- Speedup achieved through transposing kernel structures or rewriting kernels to expose more parallelism.
- Impact: Faster scramjet simulations.
- We plan to port the entire unstructured CFD solver to support GPUs.

*** Speed up results compared to single CPU core perfect linear strong scaling across 48 cores**